w.c.s. - Development #48407

pouvoir qualifier de "longs" certains afterjobs

09 novembre 2020 14:16 - Frédéric Péters

Statut:	Fermé	**Début:**	09 novembre 2020
Priorité:	Normal	**Echéance:**	
Assigné à:	Frédéric Péters	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

(mon option pour résoudre #46674, où un afterjob prend du temps et se trouve tué par uwsgi)

Pour certains jobs, je pense aux actions en masse ou à l'import de fiches depuis un csv (mais ça peut être plus fin, genre l'import de fiches depuis un csv uniquement s'il y a plus de n lignes dans le CSV), il faudrait qu'ils ne soient pas exécutés en thread démarré depuis le traitement requête/réponse d'uwsgi, mais via une mécanique avancée de uwsgi (spooler, queue, etc.) ou peut-être simplement(?) cron.

Ça demande dans tous les cas que tout l'état nécessaire soit enregistré dans le job. (alors que là on profite de locals()).

Aussi dans l'idée cron, il faudra malgré tout quelque chose qui puisse assurer l'exécution de plusieurs afterjobs en parallèle, que ça soit via async/await, multiprocessing ou autre. (histoire que quelqu'un qui lance une action de masse ne bloque pas toute la plateforme).

Demandes liées:

Lié à w.c.s. - Bug #46674: uwsgi peut interrompre un job d'action de masse	**Fermé**	**16 septembre 2020**

Révisions associées

Révision 8a5ff156 - 18 décembre 2020 15:56 - Frédéric Péters

general: use uwsgi spooler to run afterjobs (#48407)

Révision fd50f701 - 18 décembre 2020 15:56 - Frédéric Péters

backoffice: convert update digests action to afterjob class (#48407)

Révision b84ff35c - 18 décembre 2020 15:56 - Frédéric Péters

backoffice: convert csv import to afterjob class (#48407)

Révision 9079904a - 18 décembre 2020 15:56 - Frédéric Péters

backoffice: convert mass action execution to afterjob class (#48407)

Révision c692620b - 18 décembre 2020 15:56 - Frédéric Péters

backoffice: convert csv/ods exports to afterjob class (#48407)

Historique

#1 - 09 novembre 2020 14:16 - Frédéric Péters

- Lié à Bug #46674: uwsgi peut interrompre un job d'action de masse ajouté

#2 - 16 novembre 2020 10:47 - Frédéric Péters

- Sujet changé de pouvoir qualifier de "longs" certains afterjobs, et que ceux-ci se lancent via cron à pouvoir qualifier de "longs" certains afterjobs

- Description mis à jour

(sujet/description amendés pour évoquer les possibilités d'uwsgi, queue/spooler/etc.)

#3 - 16 novembre 2020 10:59 - Lauréline Guérin

juste une liste non exhaustive de pistes:
- celery

- https://pypi.org/project/procrastinate/ (PG)
- https://dramatiq.io/ (rabbitMQ, redis)
- uwsgi

#4 - 06 décembre 2020 13:22 - Frédéric Péters

- Assigné à mis à Frédéric Péters

#5 - 08 décembre 2020 14:02 - Frédéric Péters

- Fichier 0001-general-use-uwsgi-spooler-to-run-afterjobs-48407.patch ajouté

- Fichier 0002-backoffice-convert-update-digests-action-to-afterjob.patch ajouté

- Fichier 0003-backoffice-convert-csv-import-to-afterjob-class-4840.patch ajouté

- Fichier 0004-backoffice-convert-mass-action-execution-to-afterjob.patch ajouté

- Fichier 0005-backoffice-convert-csv-ods-exports-to-afterjob-class.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

0001 pose le système, il s'agit d'utiliser le spooler d'uwsgi (https://uwsgi-docs.readthedocs.io/en/latest/Spooler.html) mais la dépendance est minimisée, le code qui y est exécuté est uniquement celui-ci :

```
+@spool
+def run_after_job(args):
+    from django.conf import settings
+    subprocess.run([
+        settings.WCS_MANAGE_COMMAND,
+        'runjob',
+        '--domain', args['tenant_dir'].strip('/').split('/')[-1],
+        '--job-id', args['job_id']
+    ])
```

c'est-à-dire qu'on lui passe le tenant et un identifiant de job, et qu'il passe la main à une nouvelle commande de management (runjob) pour l'exécution.

Cette commande également est très simple,

```
+    def handle(self, *args, **options):
+        domain = options.pop('domain')
+        self.init_tenant_publisher(domain)
+        try:
+            job = AfterJob.get(options['job_id'])
+        except KeyError:
+            raise CommandError('missing job')
+        job.run()
```

Pour arriver à ça l'évolution principale est que l'objet AfterJob, à qui la fonction à exécuter était passée, peut désormais être hérité, dans une classe qui définira une méthode execute, ex:

```
+class TestAfterJob(AfterJob):
+    def execute(self):
+        pass
```

Le système storage de w.c.s. réinitialise la classe des objets chargés depuis le pickle, ça ferait perdre la classe héritée, revenir à la classe AfterJob de base et ne pas savoir quoi exécuter. Pour contrer ça, il y a désormais un attribut _reset_class sur StorableObject, qui est ici mis à False.

Pour lancer l'exécution, ça reste HTTPResponse.process_after_jobs() qui est réduit à :

```
     def process_after_jobs(self):
-        [tout le reste]
+        for job in self.after_jobs or []:
+            job.run(spool=True)
```

On reste sur la mécanique actuelle qui lancera le process_after_jobs dans un thread via AfterJobsMiddleware, pas de changement ici.

Quand spool est passé à la méthode run() (et que l'afterjob a un id et une méthode), plutôt que l'exécution, le run() fait :

```
+        if spool and self.id and self.execute:
+            from django.conf import settings
+            if 'uwsgi' in sys.modules and settings.WCS_MANAGE_COMMAND:
+                from .spooler import run_after_job
+                self.store()
```

```
+                run_after_job.spool(tenant_dir=get_publisher().app_dir, job_id=self.id)
+                return
```

qui fera monter ça dans le spooler.py vu plus haut, qui lancera le process wcs-manage runjob ..., qui lancera le code.

Dans les détails à noter sur le patch, le uwsgi.ini,

```
+spooler-processes = 3
+import = wcs.qommon.spooler
```

et l'unit systemd :

```
-ExecStart=/usr/bin/uwsgi --ini /etc/%p/uwsgi.ini
+ExecStartPre=/bin/mkdir -p /var/lib/wcs/spooler/%m/
+ExecStart=/usr/bin/uwsgi --ini /etc/%p/uwsgi.ini --spooler /var/lib/wcs/spooler/%m/
```

Le paramètre --spooler est passé via l'unit, plutôt qu'être défini dans l'uwsgi.ini, pour avoir un répertoire spécifique par machine (le %m donne le "machine id").

Voilà pour la lecture presqu'intégrale de 0001.

Les autres patchs se ressemblent, ce sont des conversions d'afterjobs actuels vers des classes héritant de AfterJob, pour prendre le premier exemple assez court de 0002 (l'exécution de .store() sur tous les formdata, après avoir changé un gabarit de résumé).

On était sur l'exécution d'une fonction, elle disparait, on mentionne à la place une classe particulière, ici UpdateDigestAfterJob. On ne peut plus profiter des variables qu'on avait dans le contexte du moment, ici le self.formdef, donc on passe ça à l'objet.

```
-            def update(job=None):
-                for formdata in self.formdef.data_class().select(order_by='id'):
-                    formdata.store()
-            job = get_response().add_after_job(N_('Updating digests'), update)
+            get_response().add_after_job(UpdateDigestAfterJob(formdef=self.formdef))
```

Ensuite, la classe elle-même :

```
+class UpdateDigestAfterJob(AfterJob):
+    label = N_('Updating digests')
+
+    def __init__(self, formdef):
+        super().__init__(formdef_class=formdef.__class__, formdef_id=formdef.id)
+
+    def execute(self):
+        formdef = self.kwargs['formdef_class'].get(self.kwargs['formdef_id'])
+        for formdata in formdef.data_class().select(order_by='id'):
+            formdata.store()
```

le truc particulier ici c'est qu'on décompose le formdef reçu, c'est parce que tout l'afterjob est picklé, et éviter d'avoir ainsi tout le formdef picklé dedans, surtout que les formdef se picklent sans l'attribut fields. Donc on décompose en classe + id, puis dans l'execute on récupère ça (self.kwargs étant un dictionnaire reprenant tout ce qui aura été passé au constructeur d'AfterJob).

0003 pour convertir la création de fiches depuis un CSV, 0004 pour convertir l'exécution en masse d'une action, 0005 pour convertir les vues d'expot CSV/ODS.

#6 - 18 décembre 2020 15:02 - Thomas Noël

0001:

- dans uwsgi.ini c'est pas plutôt spooler-python-import au lieu de juste import ?
- j'ajouterais bien un commentaire dans le uwsgi.ini qui rappelle que le répertoire spooler= est posé en ligne de commande (via wcs.service ou init.d/wcs)
- dans wcs/settings.py je completerais « # management command, used to run afterjobs in uwsgi mode » avec: « ... typically "/usr/bin/wcs-manage" » parce que je suis comme ça
- je me demande si la limite à 3 process ne va pas bloquer le 4ème job qui attendra que les 3 autres soient finis... Si oui 3 me semble un peu faible pour notre SaaS, je pousserais bien à 10 par défaut (comme le cheaper-initial). Mais on peut tourner ainsi et voir la réalité et ajuster (y'a aussi spooler-frequency qui est par défaut à 30s et je me demande à quoi ça va correspondre pour nous)

0002 à 0005: tout ok ; et c'est assez propre au final.

#7 - 18 décembre 2020 15:33 - Frédéric Péters

> dans uwsgi.ini c'est pas plutôt spooler-python-import au lieu de juste import ?

ça vient de https://uwsgi-docs.readthedocs.io/en/latest/PythonDecorators.html#example-a-django-session-cleaner-and-video-encoder

```
; load the task.py module
import = task
```

mais testé ça marche pareil avec spooler-python-import.

> j'ajouterais bien un commentaire dans le uwsgi.ini qui rappelle que le répertoire spooler= est posé en ligne de commande (via wcs.service ou init.d/wcs)

Tout à fait, ajouté.

> dans wcs/settings.py je completerais « # management command, used to run afterjobs in uwsgi mode » avec: « ... typically "/usr/bin/wcs-manage" » parce que je suis comme ça

Ajouté (s/typically/usually).

> je me demande si la limite à 3 process ne va pas bloquer le 4ème job qui attendra que les 3 autres soient finis... (...)

Oui, et il n'y a pas le côté "cheaper" de montée à la volée du nombre de processus, mais j'ai été frileux à trop monter parce que conso mémoire, notamment chez imio, j'ai préféré me dire qu'on augmenterait en fonction de ce qu'on verrait sur nos SaaS. (et comme c'est distribué, déjà ça double par rapport au nombre mentionné).

(branche wip mise à jour avec ces commentaires)

#8 - 18 décembre 2020 15:36 - Thomas Noël

- Statut changé de Solution proposée à Solution validée

Rien d'autre à dire, allons-y ainsi !

#9 - 18 décembre 2020 16:39 - Frédéric Péters

- Statut changé de Solution validée à Résolu (à déployer)

```
commit efe289e64b4579bc5214c6a6b43b860f6b8cf84e
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Mon Nov 9 09:44:43 2020 +0100

    fields: use new map marker selection widget in front (#47066)

commit 0e9e91b2db723e0bf3f5b4be5915ad198d48222f
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Mon Nov 9 16:55:44 2020 +0100

    backoffice: add options to item field for plotting choices on a map (#47066)

commit 1daf66e50c2f93a091e02647958fb5350fa4d24b
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Mon Nov 9 09:41:01 2020 +0100

    api: add endpoint for geojson data sources (#47066)
```

#10 - 18 décembre 2020 19:16 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

Fichiers

0002-backoffice-convert-update-digests-action-to-afterjob.patch	1,79 ko	08 décembre 2020	Frédéric Péters
0001-general-use-uwsgi-spooler-to-run-afterjobs-48407.patch	15,3 ko	08 décembre 2020	Frédéric Péters
0003-backoffice-convert-csv-import-to-afterjob-class-4840.patch	2,65 ko	08 décembre 2020	Frédéric Péters
0004-backoffice-convert-mass-action-execution-to-afterjob.patch	7,31 ko	08 décembre 2020	Frédéric Péters
0005-backoffice-convert-csv-ods-exports-to-afterjob-class.patch	10,5 ko	08 décembre 2020	Frédéric Péters